MySQL Reference Architectures for Massively Scalable Web Infrastructure

MySQL Best Practices for Innovating on the Web

A MySQL® Strategy White Paper

December 2010

Table of Contents

Executive Summary

Organizations around the world are constantly seeking new ways to leverage the power of the web to drive their business. They recognize their ability to compete depends on optimizing their use of technology and leveraging best practices; whether their aim is to help their customers communicate, socialize, share and locate information, entertain, or shop for goods and services.

Success or failure of new web-based initiatives is highly dependent on selecting the right architectures and technologies from the start, eliminating extensive trial and error that may result in missed market opportunities. The key requirements include those that enable a new solution to reach the market as quickly as possible, reducing costs, and delivering the best end user experience – which usually translates into very fast performance, rock solid availability and security with adaptability to change.

Any web-based service should be designed around the following:

- High availability
- High performance
- Cost-effectively scale to meet rapidly growing demands
- Open, customizable and repeatable
- Ease of use and simplicity in design to achieve fast time to market
- Protection of customer data
- Low total cost of ownership

MySQL is deployed in 9 of the top 10 most trafficked sites on the web[1] including Google, Facebook and YouTube. This gives MySQL unique insight into how to design database-driven web architectures that deliver the highest levels of scalability and availability with the lowest levels of cost, risk and complexity.

In this whitepaper, we present four Reference Architectures based on best practices developed from working with the most successful web properties on the planet.

We have selected four components common to most web properties (user authentication / session management; content management, ecommerce and analytics) and defined the optimum deployment architectures for each. The reference architectures are categorized by “small”, “medium”, “large” and “extra large” (social networking) sites, based on sizing and availability requirements appropriate to each environment.

The whitepaper concludes with recommendations on the server and storage configurations needed to support the reference architectures.

[1] http://www.alexa.com/topsites

Considerations for High Availability

A central design point for each of the reference architectures presented in this whitepaper is the strategy used to achieve high availability. While desirable, designing each of the architectures for 100%, continuous availability is simply not practical.

As illustrated in the following figure, higher degrees of availability significantly reduce downtime, and are achieved by deploying systems with greater levels of redundancy and fault-tolerance.

However, greater redundancy increases the total cost of the system due to requirements for more hardware and software, as well as demanding a larger investment in IT staff, processes, and services to deploy and manage more complex environments.

In addition, architectures delivering the very highlest levels of availability are rigidly defined, which significantly limits the opportunity to accommodate the fast pace of change typically encountered in the most dynamic web environments.

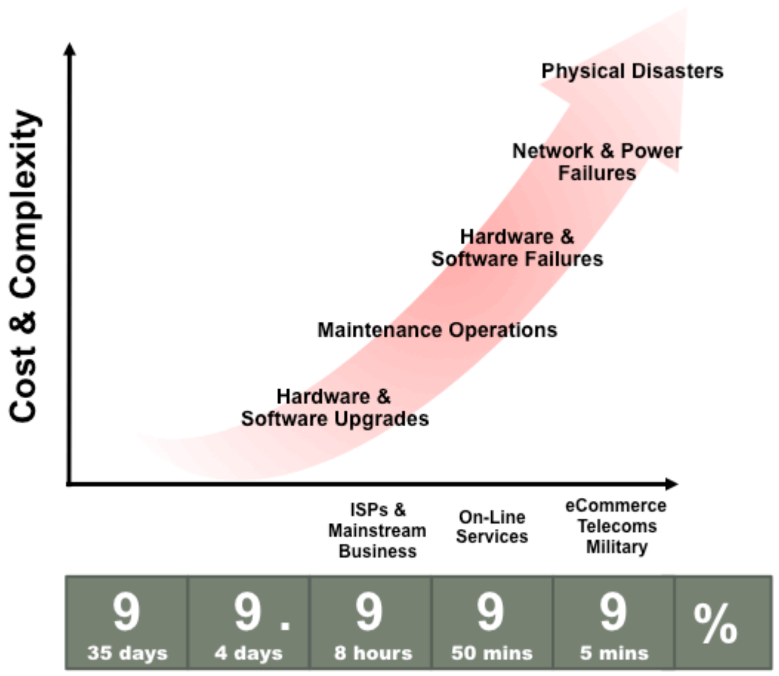


Figure 1: ***Mapping Availability to Annual Downtime***

An analysis of the business requirements for high availability and an understanding of the accompanying costs enable an optimal solution to be developed that is balanced to meet the needs of the organization within its financial and resource constraints.

As a first step, it is important to calculate the quantifiable losses of outages to the web service being deployed. It is also important to consider that direct revenue loss is only one aspect in determining the business impact caused by systems downtime. To gain a complete picture, it is also necessary to consider the following factors:
- Damage to the brand image.
- Impact to customer relationships, satisfaction and loyalty.

- Loss in employee productivity.
- Potential regulatory issues if customer or financial data is corrupted or lost.

Understanding the cost of downtime for each part of the web infrastructure is essential because this has a direct influence on the high availability technology chosen to minimize the downtime risk.

There are multiple architectures that can be used to achieve highly available database services, each differentiated by the levels of uptime they offer. These architectures can be grouped into three main categories:

- Data Replication
- Clustered & Virtualized Systems
- Shared-Nothing, Geographically-Replicated Clusters

As illustrated in the figure below, each of these architectures offers progressively higher levels of uptime, which must be balanced against potentially greater levels of cost and complexity each incurs. Simply deploying a high availability architecture is not a guarantee of actually delivering HA. In fact, a poorly implemented and maintained shared-nothing cluster could easily deliver lower levels of availability than a simple data replication solution.

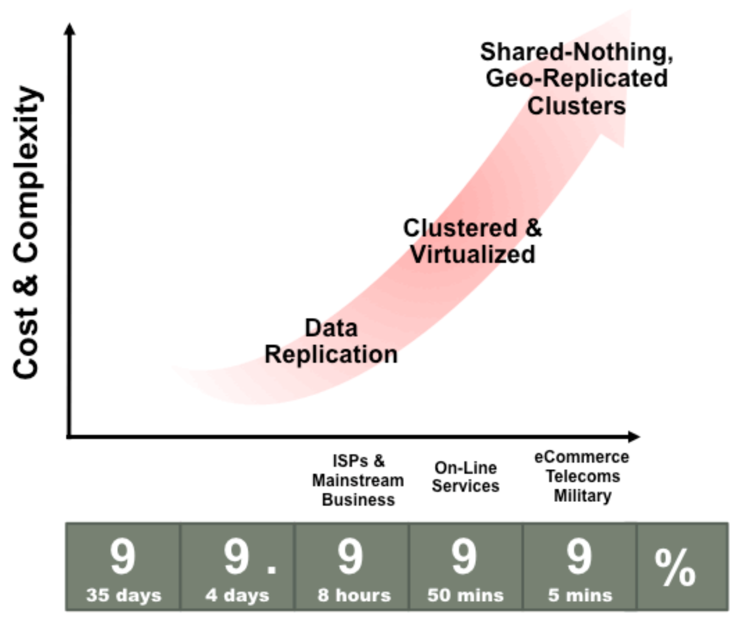
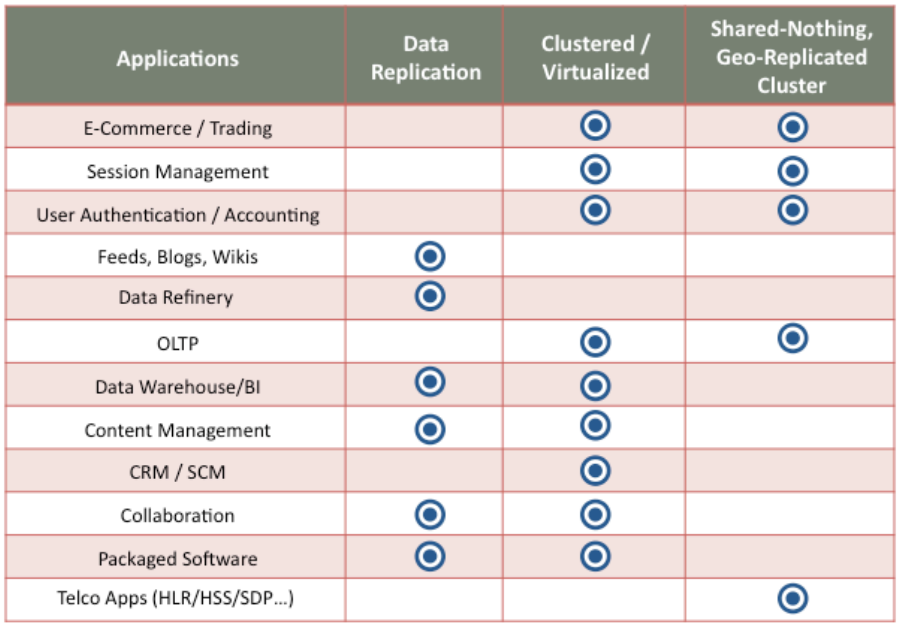


Figure 2: ***Mapping High Availability Architectures to Systems Downtime***

By understanding the availability requirements of each part of the web infrastructure it is possible to map the database to the appropriate high availability architecture.

The figure below attempts to map common application types to architectures, based on best practices observed from the MySQL user base. Of course, each organization is unique, and so while the mapping below may not be appropriate to every use-case, it does serve as a reference point to begin investigating those HA architectures which can potentially best serve your requirements.



Applications	Data Replication	Clustered / Virtualized	Shared-Nothing, Geo-Replicated Cluster
E-Commerce / Trading		◉	◉
Session Management		◉	◉
User Authentication / Accounting		◉	◉
Feeds, Blogs, Wikis	◉		
Data Refinery	◉		
OLTP		◉	◉
Data Warehouse/BI	◉	◉	
Content Management	◉	◉	
CRM / SCM		◉	
Collaboration	◉	◉	
Packaged Software	◉	◉	
Telco Apps (HLR/HSS/SDP...)			◉

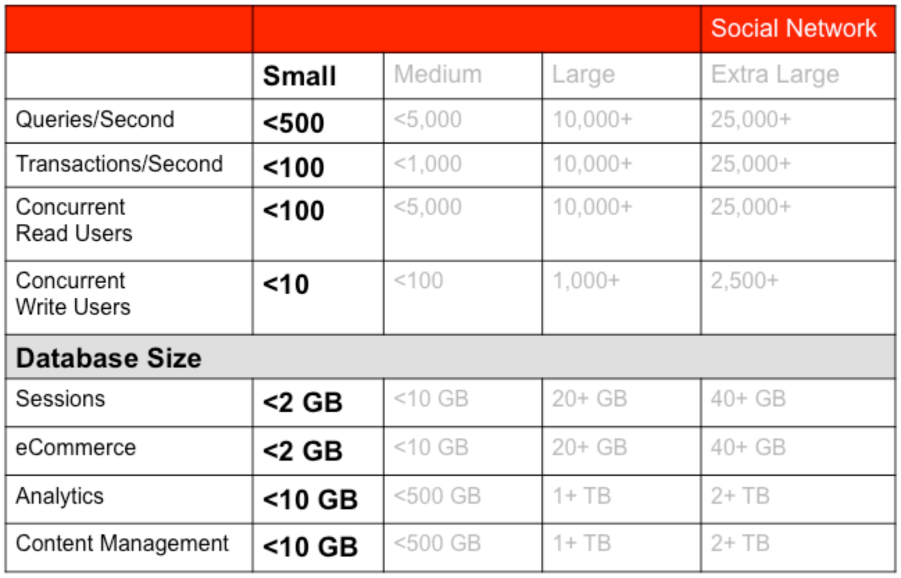
Figure 3: *Mapping Application Types to High Availability Architectures*

These best practices are captured in the Reference Architectures presented below.

Small Web Reference Architecture

Sizing & Topology

The "Small" Web reference architecture is defined by the sizing below:



				Social Network
	Small	Medium	Large	Extra Large
Queries/Second	**<500**	<5,000	10,000+	25,000+
Transactions/Second	**<100**	<1,000	10,000+	25,000+
Concurrent Read Users	**<100**	<5,000	10,000+	25,000+
Concurrent Write Users	**<10**	<100	1,000+	2,500+
Database Size				
Sessions	**<2 GB**	<10 GB	20+ GB	40+ GB
eCommerce	**<2 GB**	<10 GB	20+ GB	40+ GB
Analytics	**<10 GB**	<500 GB	1+ TB	2+ TB
Content Management	**<10 GB**	<500 GB	1+ TB	2+ TB

Figure 4: Sizing for Small Web Reference Architecture

The figure below shows the recommended topology to support the “Small” workload.

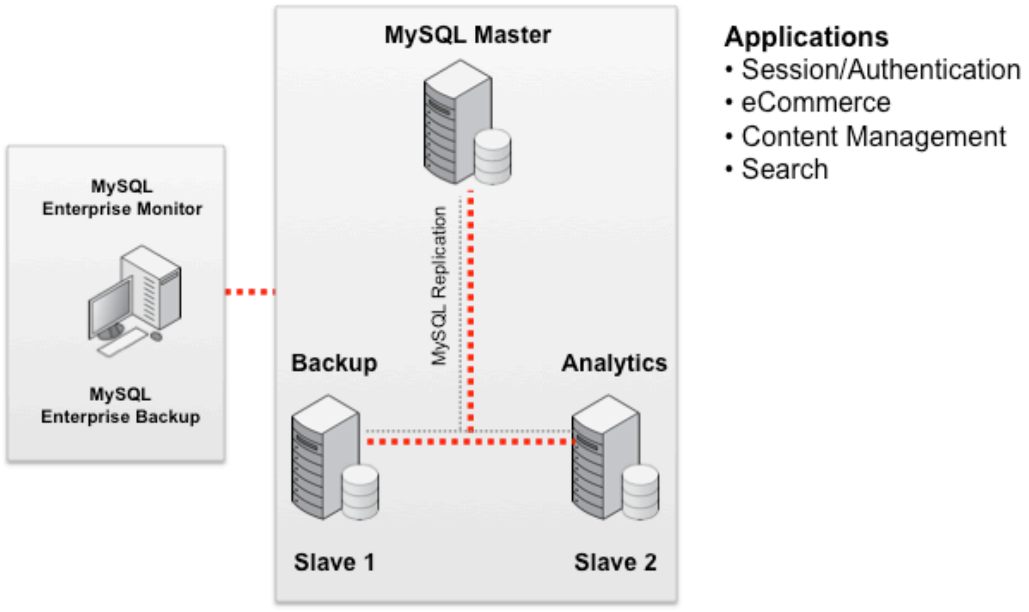


Figure 5: Topology for Small Web Reference Architecture

In this topology, a single MySQL master server is deployed to support all applications, including session management, ecommerce, content management and search.

To ensure the MySQL master is able to dedicate resources to serving the web applications, the database is replicated to two slaves; one handling backup, the other handling analytics (note: MySQL Enterprise Backup can perform online "Hot", non-blocking backups of your MySQL databases. Full backups can be performed on all InnoDB data, while MySQL is online, without interrupting queries or updates, thereby eliminating the need to use a dedicated slave).

Replication is a standard part of the MySQL database, so a replicated set of MySQL Master / Slave servers can be deployed immediately at installation.

InnoDB

The MySQL InnoDB storage engine is suitable for supporting all of the applications that are part of the Small Web Reference Architecture.

From the release of MySQL 5,5, InnoDB has become the default storage engine. InnoDB is fully ACID-compliant, offering fast, reliable recovery from crashes with zero committed data loss. InnoDB supports highly concurrent applications with row level locking and MVCC (Multi-Version Concurrency Control) support, in addition to implementing foreign keys and constraints.

Scaling the “Small” Web Reference Architecture

Common session management techniques like browser and web server-based solutions will satisfy requirements if traffic is low and the application does not need to manage large or complex session data. However as application volume and traffic increases, so do the sessions that will need to be supported on the server. Coupled with the size and complexity of the data, scalability and performance issues may also quickly arise.

Using MySQL over traditional session management solutions to store these session variables, can result in better overall performance. Therefore, if the service grows, it is recommended that the Session Management application be managed by MySQL and migrated to its own MySQL server.

As the load from the web service grows, allocating memory and tuning for each application on shared hardware resources becomes increasingly complex. It is therefore recommended that this topology should only be selected when load is light and the user expects limited growth to their service or application. As load increases, manageability becomes increasingly complex.

For the reasons above, if high growth is anticipated, it is recommended users start with the "Medium" configuration, presented later in the paper, which provides greater capacity, more flexibility for business change and improved availability.

Achieving HA – MySQL Replication

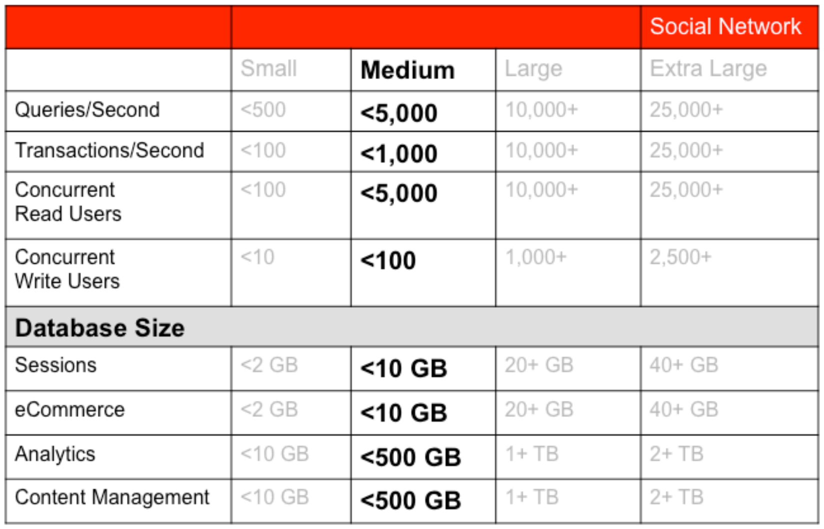
MySQL Replication has been widely deployed by some of the leading properties on the web to deliver extreme levels of database scalability. It is simple for users to rapidly create multiple replicas of their database to scale-out beyond the capacity constraints of a single instance, enabling them to serve rapidly growing database workloads.

A detailed discussion of MySQL Replication is presented in the Medium Web reference architecture section of this paper.

Medium Web Reference Architecture

Sizing & Topology

The "Medium" web reference architecture is defined by the sizing below:



				Social Network
	Small	**Medium**	Large	Extra Large
Queries/Second	<500	**<5,000**	10,000+	25,000+
Transactions/Second	<100	**<1,000**	10,000+	25,000+
Concurrent Read Users	<100	**<5,000**	10,000+	25,000+
Concurrent Write Users	<10	**<100**	1,000+	2,500+
Database Size				
Sessions	<2 GB	**<10 GB**	20+ GB	40+ GB
eCommerce	<2 GB	**<10 GB**	20+ GB	40+ GB
Analytics	<10 GB	**<500 GB**	1+ TB	2+ TB
Content Management	<10 GB	**<500 GB**	1+ TB	2+ TB

Figure 6: Sizing for Medium Web Reference Architecture

The figure below shows the recommended topology to support the “Medium” workload.

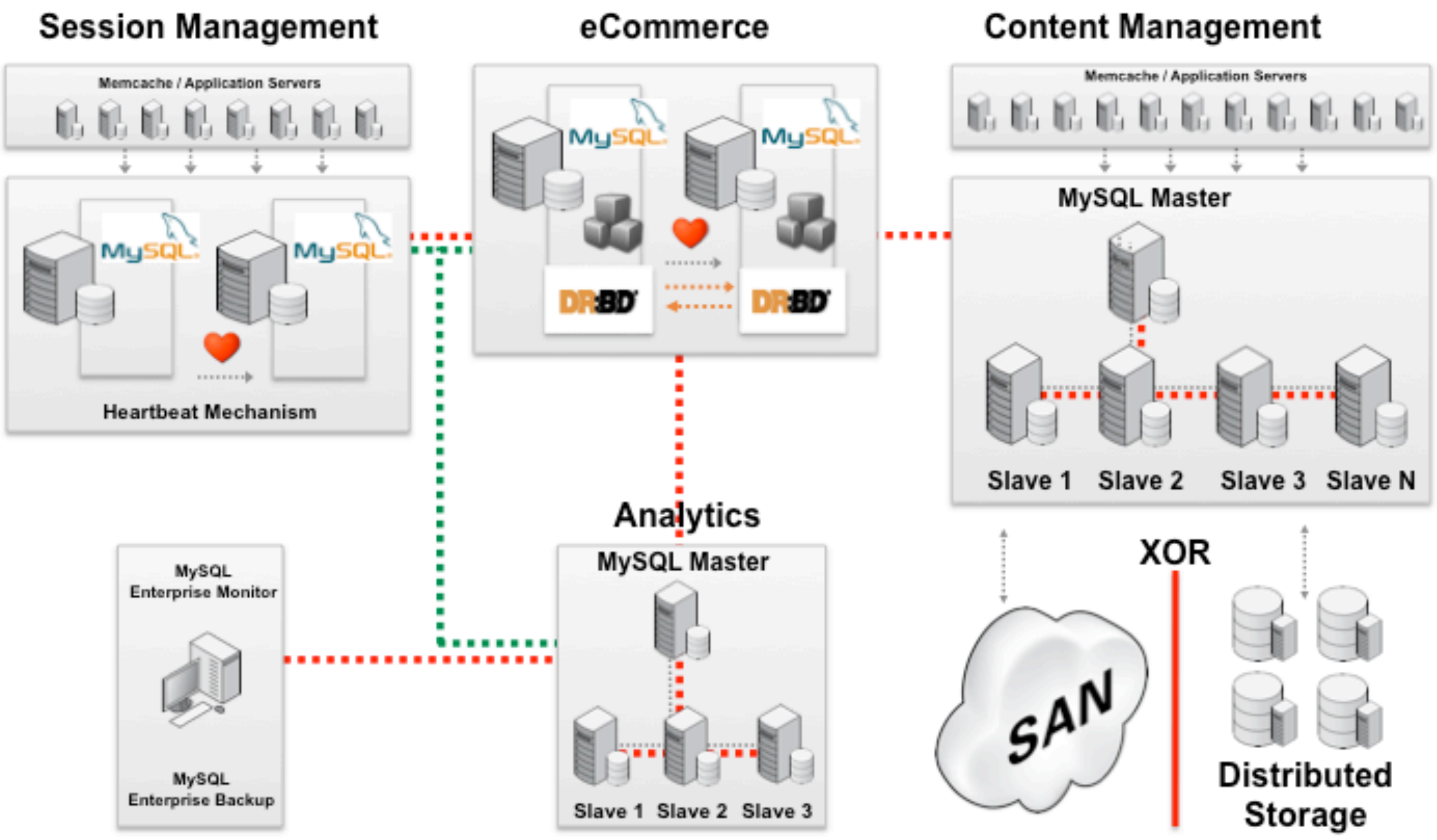


Figure 7: Topology for Medium Web Reference Architecture

Compared to the Small Web Reference Architecture described earlier, the Medium topology distributes the core functions of Session Management, Ecommerce, Content Management and Analytics across their own server and storage infrastructures, enabling each to be deployed, managed and scaled independently.

If a user expects their web service to scale over time to support the sorts of loads defined above, it is recommended they start out by initially deploying this Medium topology. This approach also provides much simpler evolution and manageability of the architecture as the workloads evolve and grow beyond the initial design and business expectations.

To size the application server to MySQL server ratio, a good rule-of-thumb is that each MySQL server will support 8 application servers, with more slaves added as application servers scale out in a read-intensive environment. For PHP applications, more application servers are generally required, and for Java, it is less.

As the connection loads to the MySQL servers increase, so do the demands to scale-out each component of the infrastructure. Independently replicated MySQL Master/Slave servers for each of the core functions affords developers and DBAs much more flexibility and control over their MySQL infrastructure as their requirements grow.

Session Management & Ecommerce
Both workloads use the default InnoDB storage engine to provide transactional support and crash recovery. To deliver high availability, both also use Linux Heartbeat and, optionally, DRBD (discussed in the "Achieving HA" section of this chapter), along with MySQL Enterprise Backup.

In typical web properties, session state is typically maintained for 45 minutes to 1 hour in a dedicated database partition, with rolling partitions used to quickly delete aged session data by dropping the affected table.

For data mining and business intelligence applications, session and ecommerce data is captured in an analytics database for off-line report generation.

As web properties seek to enhance their users' experience through personalization based on historic browsing and buying behaviors, so session data is becoming more critical. As larger volumes of session data are managed and persisted in real-time, so the workload can become increasingly performance intensive, while also demanding very high levels of availability to ensure a seamless customer experience. In these scenarios, it makes sense to swap out the InnoDB storage engine and replace it with MySQL Cluster.

As a real-time database designed for 99.999% availability, MySQL Cluster could also replace both DRBD and the Memcached layer. Again, it is worth engaging with the MySQL consulting team in Oracle to profile the application and determine the optimum selection of storage engine. MySQL Cluster is discussed in the "Large" Reference Architecture chapter of this paper.

Content Management
Scalability of the content management application is critical as this is a core part of the web service. MySQL Replication is used to deliver read scalability with each MySQL master typically attached to 20 – 30 slaves. In a regular content management workload, each slave should be able to support up to 3,000 concurrent users.

These metrics are guides only, and are highly dependent on a number of factors, including:
- Read and Write volumes
- Distribution and load balancing of web traffic
- Caching mechanisms used

Distributed file systems such as MogileFS are often used to index the physical assets of the content management system, with the metadata for each asset stored within InnoDB tables managed by MySQL. The content assets themselves – images, videos, documents, etc. – are not stored in the database, but rather within the file system.

For physical storage, the content assets can be stored either on a SAN (Storage Area Network) or distributed across local storage devices attached to each server. As a SAN can be a Single Point of Failure (SPOF), it is recommended that only mid-range and high-end products are selected as these usually include the mechanisms to deliver High Availability. The use of NAS (Network Attached Storage) or NFS (Network File Systems) is not recommended.

If the assets are to be distributed across local storage devices, it is important to ensure that the appropriate mechanisms for high availability of indexing and metadata are implemented. A solution commonly deployed to address these availability requirements is Linux Heartbeat with DRBD, which is discussed later in the chapter.

Memcached

In the Medium Reference Architecture, Memcached is deployed with MySQL to support scaling the session and content management services. Memcached is a distributed memory caching layer, and has been widely used with MySQL in some of the largest web properties to enable high levels of scalability, allowing for faster page loads and the more efficient use of existing database resources.

Once the source data is populated in the cache, future requests for the same data make use of the cached copy rather than fetching the data from the original source database, reducing load on the database layer. Memcached can cache not only data and result sets, but also, objects, such as pre-compiled HTML.

Memcached can be installed on dedicated servers or co-located with web, application or database servers. Memcached can be incrementally scaled-out in an on-demand fashion. Because memcached can scale to support dozens or even hundreds of nodes with minimal overhead, anywhere spare memory resources exist represents an opportunity to further scale an application. Memcached is designed as a non-blocking event-based server with no special networking or interconnect requirements.

Achieving HA – MySQL Replication

As introduced earlier in the paper, replication enables a database to copy or duplicate changes from one physical location or system to another (typically from the “master” to a “slave” system). This is typically used to increase the availability and scalability of a database, though as in the architecture above, users will often also perform back-up operations or run analytical queries against the slave systems, thereby offloading such functions from the master systems.

MySQL natively supports one-way, asynchronous replication as a standard feature of the database. With MySQL 5.5 and above, support for semi-synchronous replication has been added. Depending on the configuration, you can replicate all databases, selected databases, or even selected tables within a database.

MySQL replication works by simply having one server act as a master, while one or more servers act as slaves. The master server will log the changes to the database. Once these changes have been logged, they are then sent and applied to the slave(s).

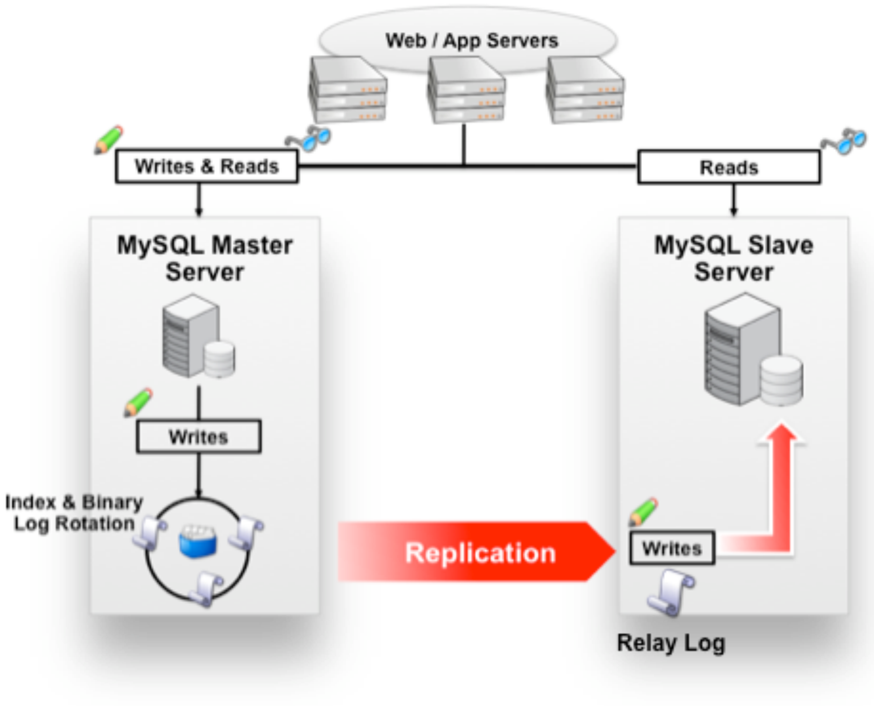


Figure 8: ***MySQL Replication Supports HA and Read Scalability "Out of the Box"***

Replication is often employed in a scale-out implementation so that requests which simply "read" data can be directed to slave servers. This allows transactions involving writes to be exclusively executed on the master server. This typically leads to not only a performance boost but a more efficient use of resources.

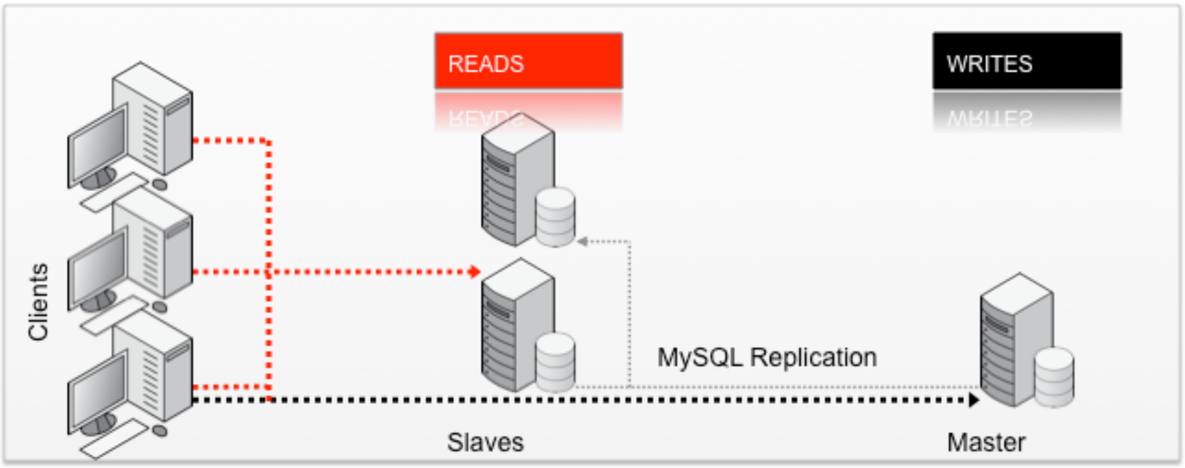


Figure 9: ***Scaling Reads with MySQL Replication***

Using MySQL replication provides the ability to scale out large web farms where reads (SELECTs) represent the majority of operations performed on the database. Slaves present very little overhead to master servers and it is not uncommon to find up to 30 slaves deployed per master in larger web properties. This is highly dependent on the traffic patterns of the workload, and so it is best to work with the MySQL Consulting team at Oracle to develop the most appropriate topology.

Replication is also the most common approach to delivering High Availability (HA) for MySQL databases. Updates are replicated from a master to slave server with the goal being to fail-over to the slave server in the event the master goes offline either due to an error, crash or for maintenance purposes. Failover can be implemented at the application or database level using a variety of mechanisms, as discussed later in the paper. Replication alone does not offer failover mechanisms. Solutions for failover are discussed later in the paper.

When MySQL is configured to use the default asynchronous replication, the master writes events to its binary log but does not know whether or when a slave has retrieved and processed them. If the master crashes, transactions that it has committed might not have been successfully replicated to any slave. Consequently, failover from master to slave in this case may result in failover to a server that is missing the most recently committed transactions.

New in MySQL 5.5 is support for semi-synchronous replication which improves data integrity, and can be used as an alternative to the built-in asynchronous replication.

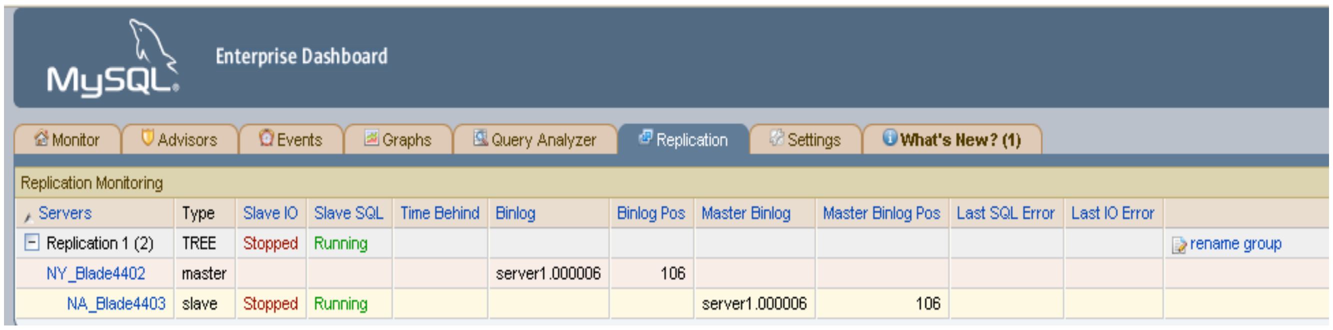
When data is replicated from the master to the slave using semi-synchronous replication, a commit is successfully returned, so it is known that the data exists in at least two places (on the master and at least one slave). A thread that performs a transaction commit on the master blocks after the commit is complete and waits until at least one semi-synchronous slave acknowledges that it has received all events for the transaction in its relay log, or until a timeout occurs. Note that the master does not have to wait until the slave has executed the statement, only that it has been received into the relay log.

Semi-synchronous replication does have some performance impact because commits are slower due to the need to wait for slaves. This is the tradeoff for increased data integrity. The amount of slowdown is at least the TCP/IP roundtrip time to send the commit to the slave and wait for the acknowledgment of receipt by the slave. This means that semi-synchronous replication works most effectively for servers that are physically co-located, communicating over fast networks.

Managing MySQL Replication

Many MySQL customers use the MySQL Enterprise Monitor, provided as part of the MySQL Enterprise and MySQL Cluster Carrier Grade Editions, with its GUI-based dashboard to monitor their replication topologies. As web properties grow and the number of masters and slaves increases, this tool has proven to save significant time and effort in managing MySQL replication.

The MySQL Enterprise Dashboard makes it easier to scale-out and achieve high availability using MySQL replication by providing advanced auto detection, grouping, documenting and monitoring of all MySQL replication master/slave hierarchical relationships. Changes and additions to existing replication topologies are also auto-detected and maintained, providing DBAs instant visibility into newly implemented updates. This helps reduce the learning curve for DBAs new to MySQL replication or to specific high availability and scale-out environments.

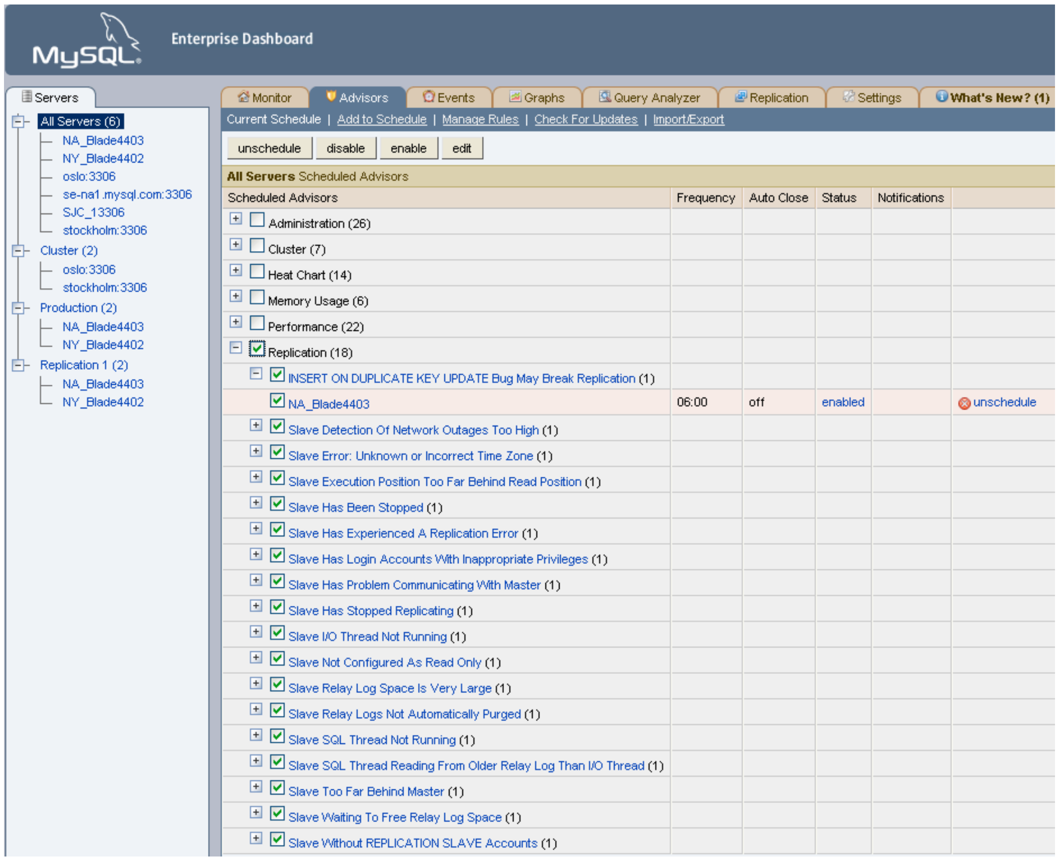


Replication Monitoring

Servers	Type	Slave IO	Slave SQL	Time Behind	Binlog	Binlog Pos	Master Binlog	Master Binlog Pos	Last SQL Error	Last IO Error	
Replication 1 (2)	TREE	Stopped	Running								rename group
NY_Blade4402	master				server1.000006	106					
NA_Blade4403	slave	Stopped	Running				server1.000006	106			

Figure 10: *MySQL Enterprise Monitor - Replication Dashboard*

The Replication Monitor provides a consolidated, real-time view into the health, performance and availability of all master/slave topologies. Working with the Replication Advisor Rules, the Replication Monitor helps the DBA to proactively identify and correct replication-related problems before they can become costly outages.



All Servers (6)
- NA_Blade4403
- NY_Blade4402
- oslo:3306
- se-na1.mysql.com:3306
- SJC_13306
- stockholm:3306

Cluster (2)
- oslo:3306
- stockholm:3306

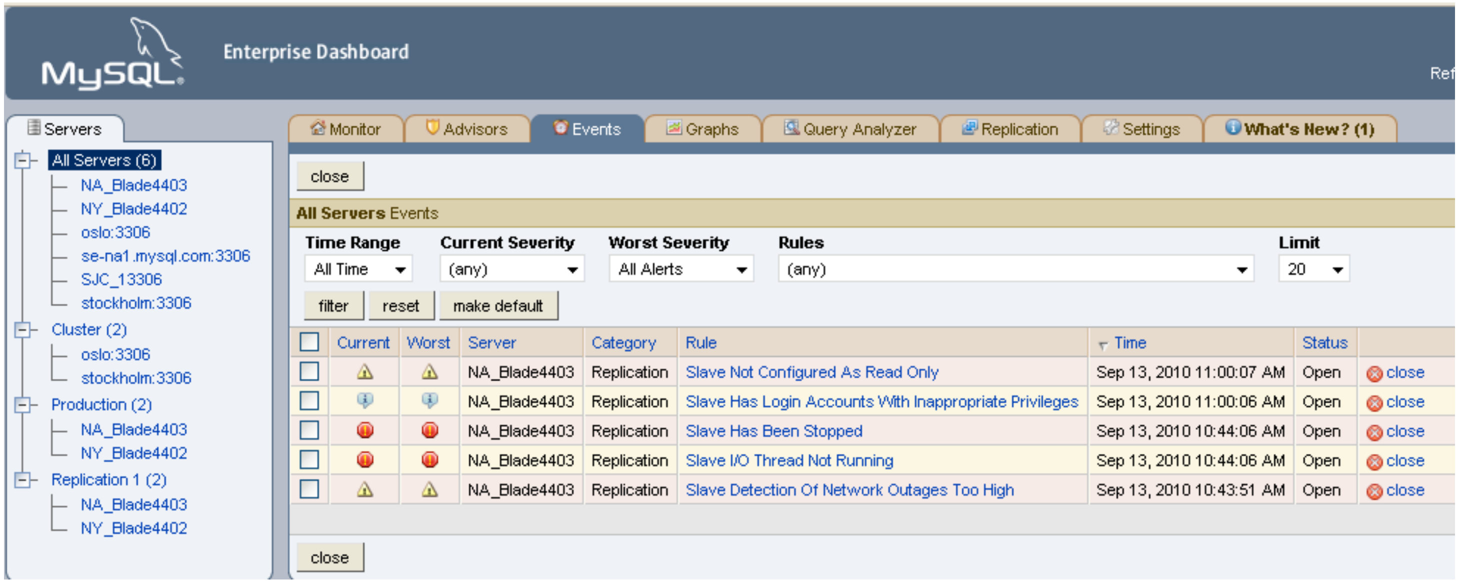
Production (2)
- NA_Blade4403
- NY_Blade4402

Replication 1 (2)
- NA_Blade4403
- NY_Blade4402

All Servers Scheduled Advisors

Scheduled Advisors	Frequency	Auto Close	Status	Notifications	
Administration (26)					
Cluster (7)					
Heat Chart (14)					
Memory Usage (6)					
Performance (22)					
Replication (18)					
INSERT ON DUPLICATE KEY UPDATE Bug May Break Replication (1)					
NA_Blade4403	06:00	off	enabled		unschedule
Slave Detection Of Network Outages Too High (1)					
Slave Error: Unknown or Incorrect Time Zone (1)					
Slave Execution Position Too Far Behind Read Position (1)					
Slave Has Been Stopped (1)					
Slave Has Experienced A Replication Error (1)					
Slave Has Login Accounts With Inappropriate Privileges (1)					
Slave Has Problem Communicating With Master (1)					
Slave Has Stopped Replicating (1)					
Slave I/O Thread Not Running (1)					
Slave Not Configured As Read Only (1)					
Slave Relay Log Space Is Very Large (1)					
Slave Relay Logs Not Automatically Purged (1)					
Slave SQL Thread Not Running (1)					
Slave SQL Thread Reading From Older Relay Log Than I/O Thread (1)					
Slave Too Far Behind Master (1)					
Slave Waiting To Free Relay Log Space (1)					
Slave Without REPLICATION SLAVE Accounts (1)					

Figure 11: *MySQL Enterprise Monitor - Replication Advisor*



All Servers Events

Time Range: All Time | Current Severity: (any) | Worst Severity: All Alerts | Rules: (any) | Limit: 20

	Current	Worst	Server	Category	Rule	Time	Status	
☐	⚠	⚠	NA_Blade4403	Replication	Slave Not Configured As Read Only	Sep 13, 2010 11:00:07 AM	Open	close
☐	ℹ	ℹ	NA_Blade4403	Replication	Slave Has Login Accounts With Inappropriate Privileges	Sep 13, 2010 11:00:06 AM	Open	close
☐	●	●	NA_Blade4403	Replication	Slave Has Been Stopped	Sep 13, 2010 10:44:06 AM	Open	close
☐	●	●	NA_Blade4403	Replication	Slave I/O Thread Not Running	Sep 13, 2010 10:44:06 AM	Open	close
☐	⚠	⚠	NA_Blade4403	Replication	Slave Detection Of Network Outages Too High	Sep 13, 2010 10:43:51 AM	Open	close

Figure 12: ***MySQL Enterprise Monitor - Replication Events***

As the Replication Advisor identifies a problem and sends out an alert, the DBA can use the alert content along with the Replication Monitor to drill into the status of the affected master and/or slave. Using the Replication Monitor and the expert advice from the Replication Advisor they can review the current master/slave status and view metrics (such as Slave I/O, Slave SQL thread, seconds behind master, master binlog position, last error, etc.) that are relevant to diagnosing and correcting any problems. The Replication Monitor is designed and implemented to save DBAs time writing and maintaining scripts that collect, consolidate and monitor similar MySQL replication status and diagnostic data.

Linux Heartbeat

One of the most common solutions to address failure detection and failover across a cluster of server(s) is Linux Heartbeat (for cluster messaging) and a cluster resource manager such as Pacemaker or OpenAIS. Linux Heartbeat is not the only solution for automating failover. Solaris Cluster is widely deployed in environments running the Solaris Operating System. For the purpose of these Reference Architectures, we focus on Linux Heartbeat.

Linux Heartbeat implements a heartbeat-protocol that sends messages at regular intervals between two or more nodes. If a message is not received from a node within a given interval, then it is assumed the node has failed and a failover action is initiated by the cluster resource manager.

When Heartbeat and the resource manager are initially configured on the hosts, a master node is selected. When the heartbeat starts up on the master node, a virtual IP address is assigned to the master's network interface. This interface will be the mechanism in which external processes, applications and users will access the node.

In the event of a failure, the resources on the failed host are disabled, and the resources on one of the replacement hosts are enabled instead. In addition, the Virtual IP (VIP) address for the cluster is redirected to the new host in place of the failed device.

The figure below illustrates the failover of a master MySQL server with Linux Heartbeat and Pacemaker automatically redirecting the application to the slave server.

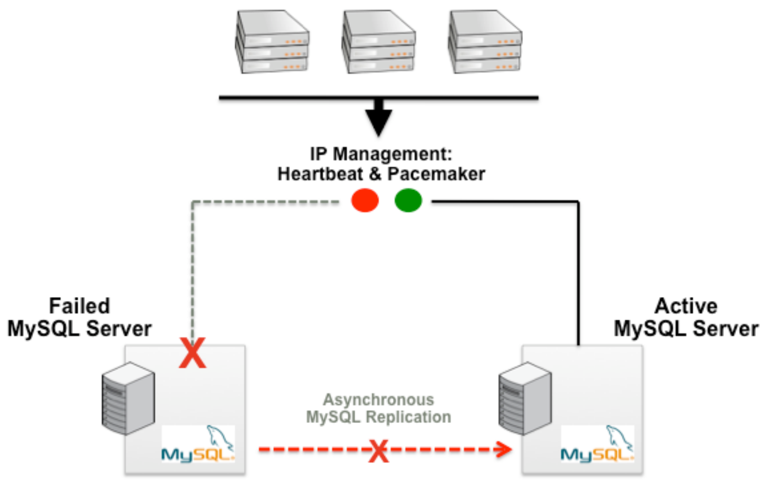


Figure 13: ***MySQL Master Failover with Heartbeat & Pacemaker***

To create a fully transaction-safe “hot standby” configuration, some users also deploy DRBD (Distributed Replicated Block Device) with Heartbeat.

DRBD is an open source Linux kernel block device which leverages synchronous replication to mirror data between two systems, typically in an Active / Passive configuration.

As illustrated in the figure below, MySQL is configured to store information on the DRBD block device, with one server acting as the primary and a second host available to operate as an immediate replacement in the event of a failure. Heartbeat manages the interfaces on the two servers and automatically configures the secondary (passive) server to replace the primary (active) server in the event of a failure.

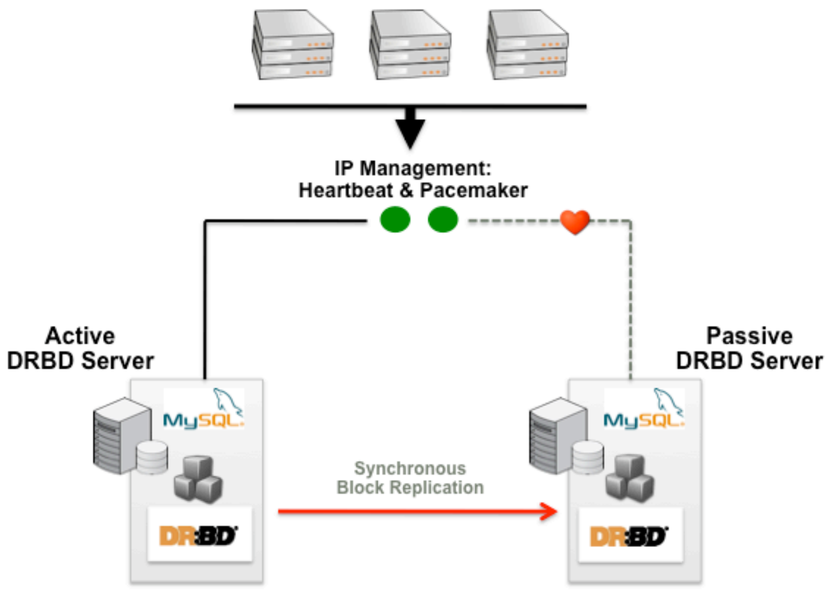


Figure 14: *Active / Passive DRBD Cluster*

Large Web Reference Architecture

Sizing & Topology

The "Large" web reference architecture is defined by the sizing below:

				Social Network
	Small	Medium	**Large**	Extra Large
Queries/Second	<500	<5,000	**10,000+**	25,000+
Transactions/Second	<100	<1,000	**10,000+**	25,000+
Concurrent Read Users	<100	<5,000	**10,000+**	25,000+
Concurrent Write Users	<10	<100	**1,000+**	2,500+
Database Size				
Sessions	<2 GB	<10 GB	**20+ GB**	40+ GB
eCommerce	<2 GB	<10 GB	**20+ GB**	40+ GB
Analytics	<10 GB	<500 GB	**1+ TB**	2+ TB
Content Management	<10 GB	<500 GB	**1+ TB**	2+ TB

Figure 15: Sizing for Large Web Reference Architecture

The figures below show both a conceptual and detailed view of the recommended topology to support the "Large" workload.

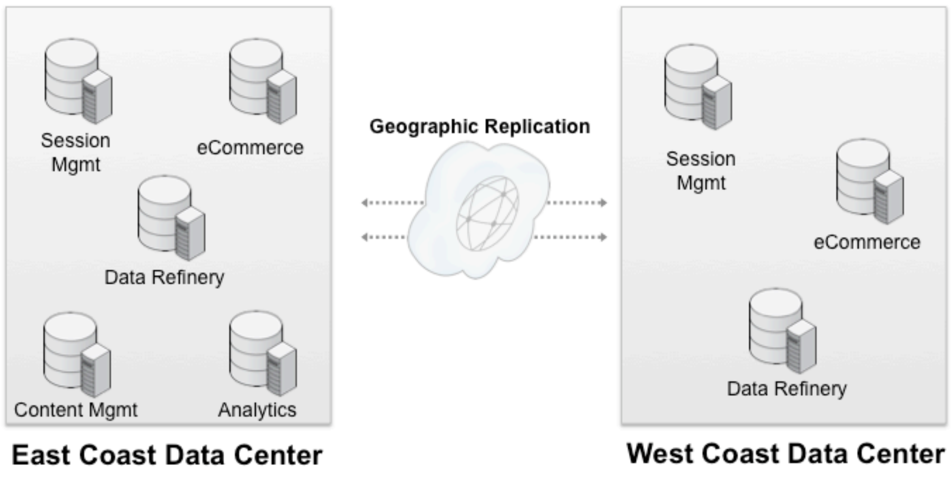


Figure 16: Conceptual View - Large Web Reference Architecture

In the conceptual view, we can see that MySQL databases are mirrored to a remote data center using MySQL Geographic Replication, therefore providing disaster recovery capabilities. Geographic replication is also commonly deployed to physically locate data closer to users in order to reduce the affects of geographic latency.

Typically the session management and ecommerce databases would be the prime candidates for replication. IP management could determine the user's location and route them to the data center that was physically closest to them. The Data Refinery would also be typically replicated.

The Large Reference Architecture builds upon the best practices defined for the Medium Reference Architecture. Master / Slave replication is used to scale-out the Content Management and Analytics components of the web property. The content management architecture would be able to scale to 100k+ concurrent users with around 30 slaves – again dependent on traffic patterns of the workload.
The content management system would be implemented either on a SAN or distributed local storage. Please refer back to the "Content Management" section of the Medium Web Reference Architecture for a discussion on the technology considerations that need to be made when evaluating the best deployment platform.

Data Refinery
In addition to the existing workloads, a Data Refinery component is added for higher volume analytics and increased content management demands.

As with the Medium-sized Reference Architecture, session management and ecommerce data are used for analysis, but in this instance, they are first replicated to the Data Refinery. The Data Refinery aggregates data from other parts of the web infrastructure, including the content management system where it performs data cleansing tasks to modify or delete dirty data. It then builds complete data sets and data warehouse dimensions for loading into the Analytics database,

The Data Refinery can also be used to cleanse data before it is loaded into the content management system.

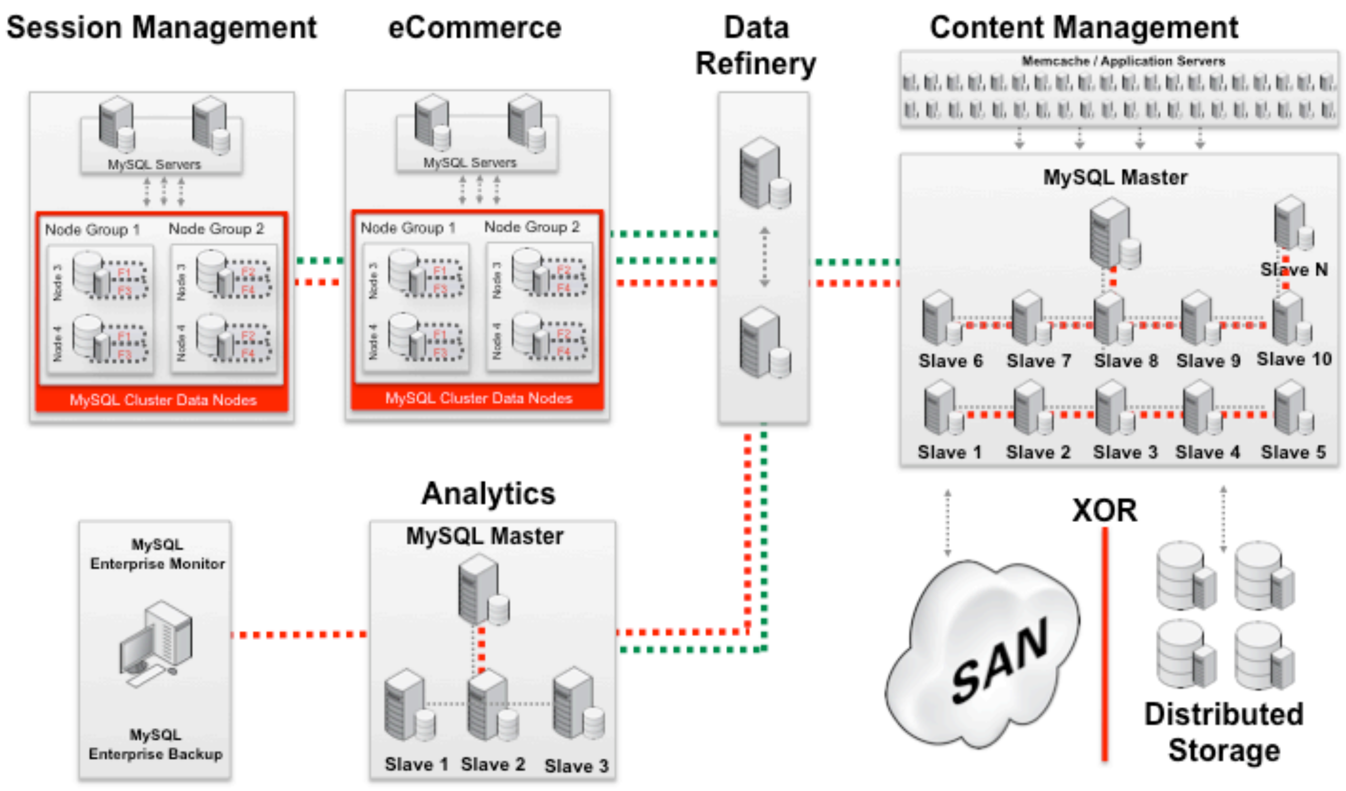


Figure 17: Detailed View - Large Web Reference Architecture

MySQL Cluster for Session Management and Ecommerce

To handle the higher performance and availability demands of the “Large” web workloads, the session management and ecommerce databases are deployed on MySQL Cluster. With 4 x data nodes, it is possible to support 6,000 sessions (page hits) a second, with each page hit generating 8 – 12 database operations.

By using the scale-out capabilities of MySQL Cluster, it is possible to combine the session management and ecommerce databases onto one larger cluster.

MySQL Cluster is a real-time, transactional data store combining the flexibility of a highly available relational database with the low TCO of open source. Featuring a “shared-nothing” distributed architecture with no single point of failure, MySQL Cluster is designed to deliver 99.999% availability demanded by transactional web environments.

MySQL Cluster's real-time design delivers predictable, millisecond response times with the ability to service tens of thousands of transactions per second. Support for in-memory and disk based data, automatic data partitioning with load balancing and the ability to add nodes to a running cluster with zero downtime allows linear database scalability to handle the most unpredictable web-based workloads.

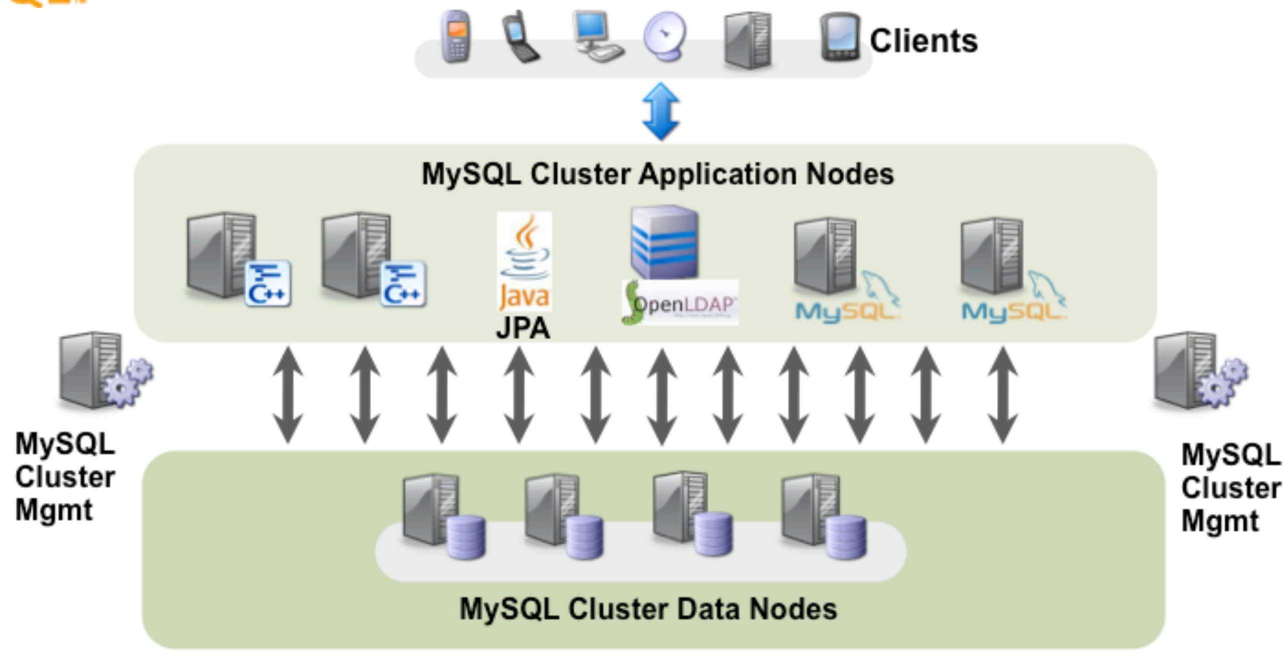


Figure 18: The MySQL Cluster architecture eliminates any single point of failure

MySQL Cluster is scaled-out on low cost, commodity servers and consists of three different types of nodes, each providing specialized services to scale web applications.

Data Nodes are the main nodes of the cluster, providing the following functionality:
• Data storage and management of both in-memory and disk-based data
• Automatic and user defined partitioning of data, even as new nodes are added to a running cluster
• Synchronous replication of data between data nodes using a 2-Phase commit protocol
• Transactions and data retrieval
• Automatic fail-over
• Resynchronization after failure

By distributing data in a shared-nothing architecture, and implementing synchronous replication with a 2-Phase commit protocol, if a Data Node happens to fail, there will always at least one additional Data Node storing the same information. This allows for requests and transactions to continue to be satisfied without interruption. Data Nodes can also be added on-line, allowing for rapid scalability of data capacity and processing, without impacting the availability of the applications.

Application Nodes are responsible for connecting applications to the data nodes. This can take the form of an application embedding the high performance NDB (C++) or Java APIs, or more commonly, they connect to MySQL Servers which provide an SQL interface into the data stored within the cluster. Thus, multiple applications can simultaneously access the data in MySQL Cluster using a rich set of interfaces with any updates to the underlying data instantly accessible to any application accessing the cluster. Application Nodes can also be added online without disrupting service from the cluster.

Management Nodes are responsible for managing the cluster and make cluster configuration information available to other nodes. The Management Nodes are used at startup and when there is a system reconfiguration. Management Nodes can be stopped and restarted without affecting the ongoing execution of the Data and Application Nodes.

By default, the Management Nodes also provides arbitration services, in the event of a network failure leading to a “split-brain”, or a cluster exhibiting “network-partitioning”.

With this distributed architecture, where dependencies have been minimized, applications continue to run and data remains consistent, even if any one of the Data, Application, or Management Nodes fail.

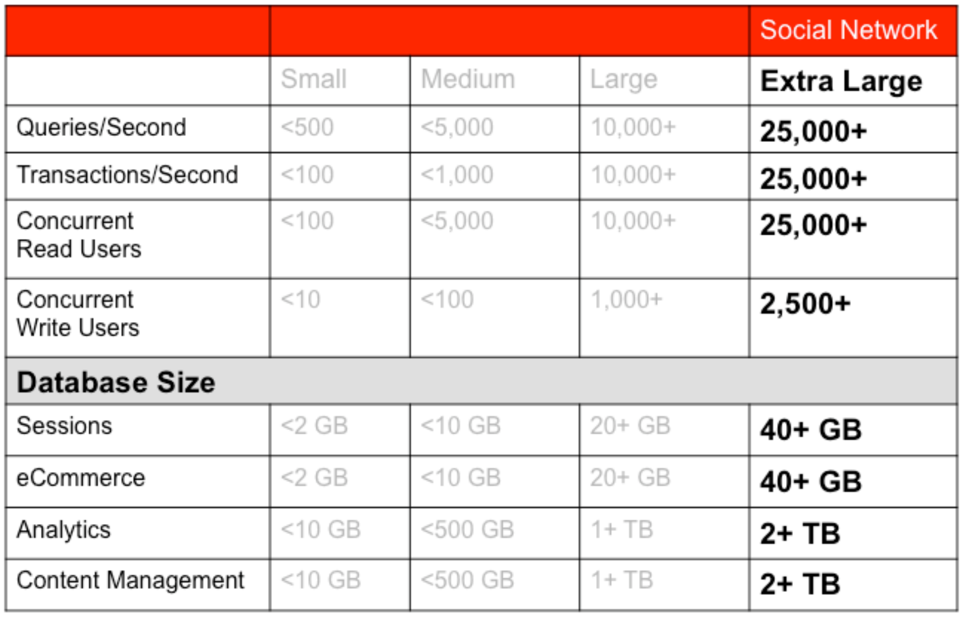
MySQL Cluster Carrier Grade Edition inclues the MySQL Cluster Manager to simplify the creation and management of the cluster by automating common management tasks. More information is in the “Value-Added Components” section of the whitepaper.

To learn more about the MySQL Cluster architecture, refer to the MySQL Cluster Architecture and New Features whitepaper posted at:
http://www.mysql.com/why-mysql/white-papers/mysql_wp_cluster7_architecture.php

Social Networking (Extra Large) Reference Architecture

Sizing & Topology

The “Social Network” reference architecture is defined by the sizing below:



				Social Network
	Small	Medium	Large	**Extra Large**
Queries/Second	<500	<5,000	10,000+	**25,000+**
Transactions/Second	<100	<1,000	10,000+	**25,000+**
Concurrent Read Users	<100	<5,000	10,000+	**25,000+**
Concurrent Write Users	<10	<100	1,000+	**2,500+**
Database Size				
Sessions	<2 GB	<10 GB	20+ GB	**40+ GB**
eCommerce	<2 GB	<10 GB	20+ GB	**40+ GB**
Analytics	<10 GB	<500 GB	1+ TB	**2+ TB**
Content Management	<10 GB	<500 GB	1+ TB	**2+ TB**

Figure 19: Sizing for Social Networking Reference Architecture

The figure below shows the recommended topology to support the “Social Networking” workload.

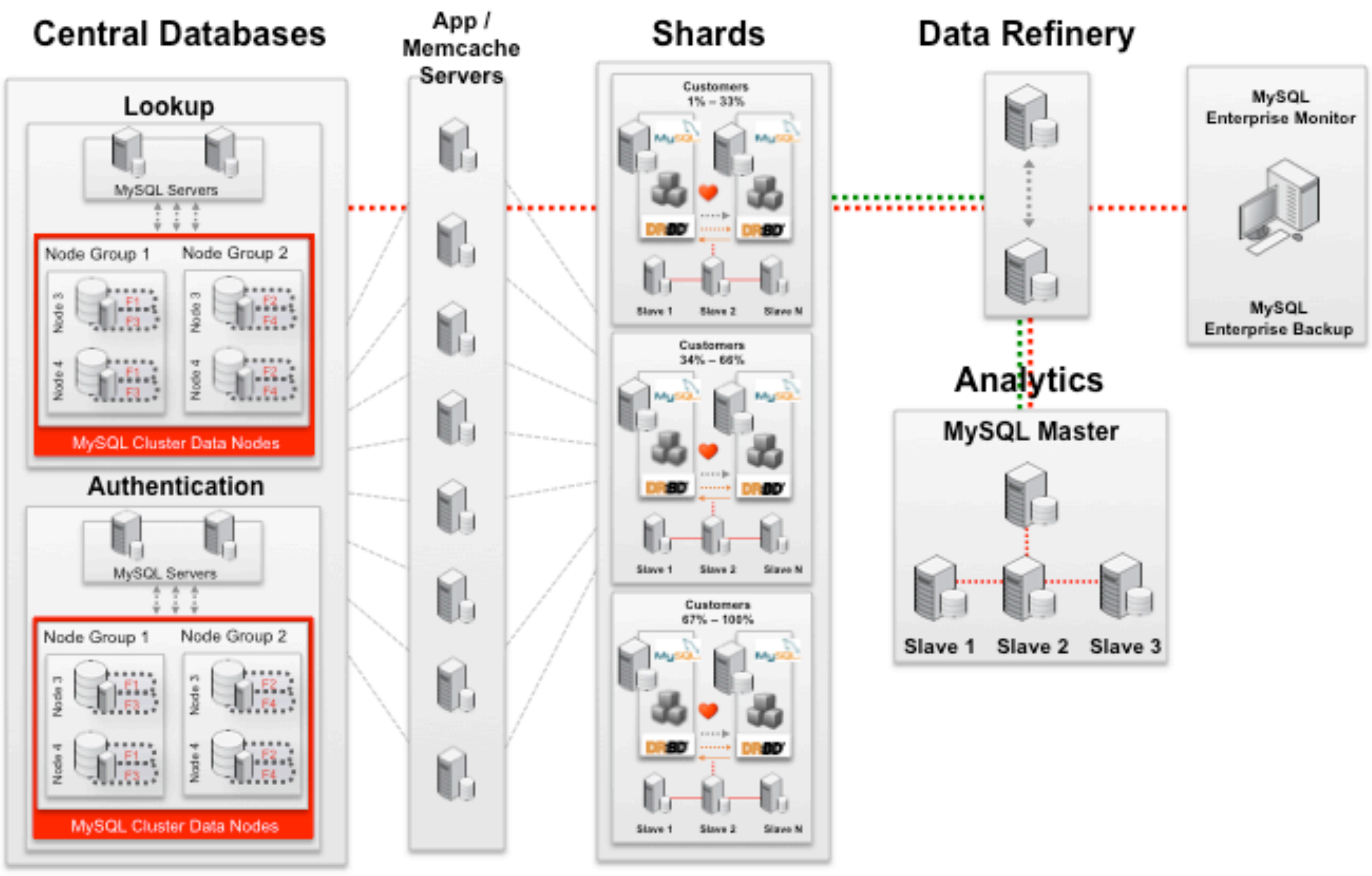


Figure 20: Topology for Social Networking Reference Architecture

The Social Networking reference architecture re-uses many concepts defined in the "Medium" and "Large" reference architectures including dedicated hardware for different workloads within the infrastructure, memcached to relieve load on the database servers and a Data Refinery for data cleansing, aggregation, etc.

MySQL Cluster is used for authentication of users and to provide the look-up mechanism used by applications to direct reads and writes to their appropriate partition or "shard" of the user database when more than one key is used for look-ups.

Sharding

Sharding is commonly used by large web properties to increase the scalability of writes across their database. In high traffic Web 2.0 environments such as social networking much of the content is generated by users themselves, thereby demanding high levels of write scalability, handling rapidly changing data. It must be remembered that reads will still predominate in Web 2.0-type environments as typically each record will be read before an update is applied.

It is also important to emphasize that millions of users can still be serviced with web applications, without requiring any database sharding at all.

Sharding – also called application partitioning - involves the application dividing the database into smaller data sets and distributing them across multiple servers. This approach enables very cost effective scalability as low cost commodity servers can be easily deployed when capacity requirements grow. In addition, query processing affects only a smaller sub-set of the data, thereby boosting performance.

Applications need to become “shard-aware” so that writes can be directed to the correct shard, and JOIN operations are minimized. If best practices are applied, then sharding becomes an effective way to scale relational databases supporting web-based write-intensive workloads.

Hashing against a single column (key) proves the most effective means to shard data. In the Social Networking Reference Architecture we have sharded the user database by User ID, so Shard One handles the first third of users, the second shard handles the second third of users, and the remainder are stored in the third shard.

This approach initially provides the most logical sharding mechanism and makes it simple to scale-out as the service grows. However, it is likely that some shards will contain more active users than other shards, meaning re-balancing of shards may be required in the future. Both the application and the database need to be sufficiently flexible to accommodate change in the architecture.

In the Reference Architecture, high availability for each shard is delivered by DRBD with Heartbeat, and each master server replicates to slaves for read scalability. Unlike the Web reference architectures, the Social Networking architecture manages session state on the shards themselves.

As sharding is implemented at the application layer, it is important to utilize best practices in application and database design. The Architecture and Design Consulting Engagement offered by the MySQL Consulting group in Oracle has enabled many customers to reduce the complexity of deployments where sharding is used for scalability.

By deploying MySQL with sharding and replication, leading social networking sites such as Facebook, Flickr, LinkedIn and Twitter have been able to scale their requirements for relational data management exponentially.

The Perfect MySQL Server

MySQL is supported across 20 different platforms, comprising multiple CPU architectures and operating systems. The following section identifies the most common platforms that have proven to deliver optimum performance, scalability and availability for web-based architectures.

MySQL Servers

When sizing the server, it is important to profile the size and characteristics of the databases you will be running now, and estimate future sizing requirements as the workload scales. It is also important to remember that slave servers used for read scalability should be as powerful as those deployed as masters. The recommended specification is as follows:

- 8 – 16 x86-64 bit CPU cores (MySQL 5.5 and above).
- 4 – 8 x86 -64 bit CPU cores (MySQL 5.1 and earlier).
- 3 – 10x more RAM than active data.
- Linux, Solaris or Windows operating systems.
- Minimum of 4 x hard disk drives. 8 – 16 disks will increase performance for I/O intensive applications.
- Hardware RAID with battery-backed cache.
- RAID 10 recommended. RAID 5 is suitable if the workload is read-intensive.
- 2 x Network Interface Cards and 2 x Power Supply Units for redundancy.

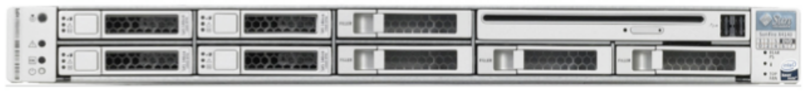


Figure 21: The Oracle Sun Fire x4170

MySQL Cluster

Due to the distributed nature of MySQL Cluster, the recommended server configurations are slightly different than a MySQL server running the MyISAM or InnoDB storage engines.

Application Nodes (MySQL Servers) Recommendation

- 4 - 16 x86-64 bit CPU cores
- Minimum 4GB of RAM. Memory is not as critical at this layer, and requirements will be influenced by connections and buffers.
- 2 x Network Interface Cards and 2 x Power Supply Units for redundancy.

Data Nodes Recommendation

- 8 x86-64 bit CPU cores. Use as high a frequency as possible as this will enable faster processing of messages
- RAM per Server = Database Size * Replicas * 1.25 (data redundancy + indexes drive overall memory requirement) / number of data nodes.
  - Example: 10GB database * 2 replicas * 1.25 / 2 data nodes = 12.5GB of RAM per data node.
- Linux, Solaris or Windows operating systems.
- 2 x Network Interface Cards and 2 x Power Supply Units for redundancy.
- Disk-drive recommendations as follows.

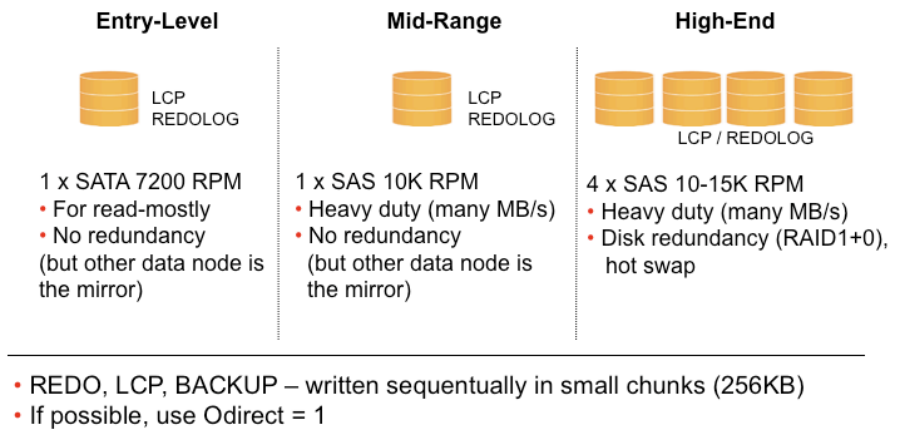

Entry-Level
Mid-Range
High-End
LCP
REDOLOG
LCP
REDOLOG
LCP / REDOLOG
1 x SATA 7200 RPM
• For read-mostly
• No redundancy (but other data node is the mirror)
1 x SAS 10K RPM
• Heavy duty (many MB/s)
• No redundancy (but other data node is the mirror)
4 x SAS 10-15K RPM
• Heavy duty (many MB/s)
• Disk redundancy (RAID1+0), hot swap
• REDO, LCP, BACKUP – written sequentially in small chunks (256KB)
• If possible, use Odirect = 1

Figure 22: Recommended Disk Drive Configurations for Checkpointing & REDO Logs

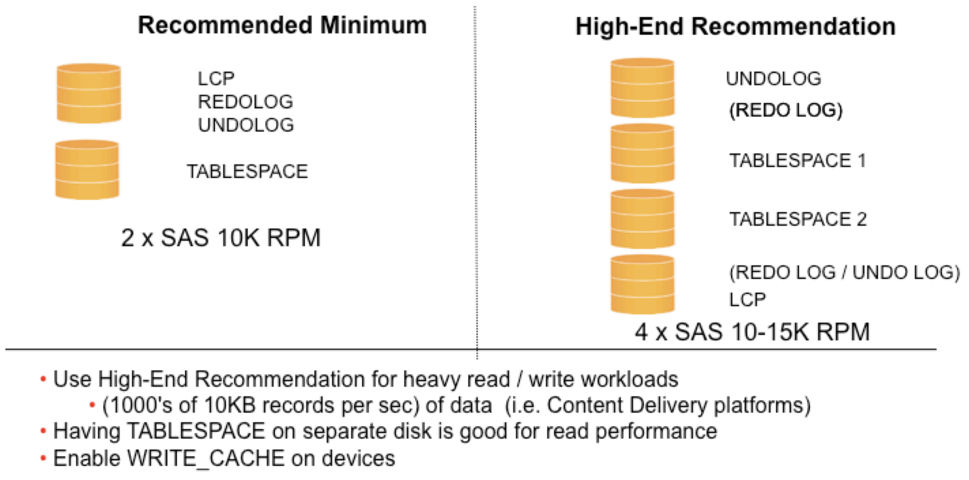

Recommended Minimum
LCP
REDOLOG
UNDOLOG
TABLESPACE
2 x SAS 10K RPM
High-End Recommendation
UNDOLOG
(REDO LOG)
TABLESPACE 1
TABLESPACE 2
(REDO LOG / UNDO LOG)
LCP
4 x SAS 10-15K RPM
• Use High-End Recommendation for heavy read / write workloads
• (1000's of 10KB records per sec) of data (i.e. Content Delivery platforms)
• Having TABLESPACE on separate disk is good for read performance
• Enable WRITE_CACHE on devices

Figure 23: Recommended Disk Drive Configurations for Disk-Based Tables

It is recommended to deploy the data and application nodes on a dedicated network (i.e. IP addresses starting at 10.0.1.0), using 1 Gigabit Ethernet as a minimum transport. The MySQL servers would then also be attached to the public network.

To provide resilience to network failures, it is recommended to configure each server with bonded and redundant Network Interface Cards (NICs), connected to redundant switches. If multiple data nodes are deployed onto a single host (i.e. Sun Oracle CMT systems), it is recommended to bond four or more NICs together to handle the increased network bandwidth requirements.

Value-Added Components of the Reference Architectures

The Reference Architectures defined above include value-added components delivered as part of the commercial MySQL offerings. These are discussed as follows.

"Only use an open-source DBMS engine supplied by a vendor who controls or participates in the engineering of the DBMS and always purchase subscription support when used in production environments."

Gartner, 2008

MySQL Enterprise Monitor

The MySQL Enterprise Monitor continuously monitors your MySQL servers and alerts you to potential problems before they impact your system. It is like having a "Virtual DBA Assistant" at your side to recommend best practices to eliminate security vulnerabilities, improve replication, optimize performance and more. As a result, the productivity of your developers, DBAs and System Administrators is improved significantly.

Plus the MySQL Query Analyzer helps you improve application performance by monitoring query performance and accurately pinpointing SQL code that is causing a slow down. Queries are presented in an aggregated view across all MySQL servers so you can filter for specific query problems and analyze your most expensive code. With the MySQL Query Analyzer, you can improve the SQL code during active development, and continuously monitor and tune the queries in production.

MySQL Enterprise BackUp

MySQL Enterprise Backup performs online "Hot", non-blocking backups of your MySQL databases. Full backups can be performed on all InnoDB data, while MySQL is online, without interrupting queries or updates. In addition, incremental backups are supported where only data that has changed from a previous backup operation is backed up. Also partial backups are supported when only certain tables or tablespaces need to be backed up.

MySQL Enterprise Backup restores your data from a full backup with full backward compatibility. Consistent Point-in-Time Recovery (PITR) enables DBAs to perform a restore to a specific point in time. Using MySQL backups and binlog, DBAs can also perform fine-grained roll forward recovery to a specific transaction. A partial restore allows recovery of targeted tables or tablespaces. Plus, DBAs can restore backups to a separate location, or create clones for fast replication setup or administration.

MySQL Enterprise Backup supports creating compressed backup files, typically reducing backup size from 70% to over 90% when compared of the size of actual database files - reducing storage and other costs.

MySQL Cluster Manager

MySQL Cluster Manager simplifies the creation and management of MySQL Cluster Carrier Grade Edition by automating common management tasks. As a result, Database Administrators (DBAs) and Systems Administrator are more productive, enabling them to focus on strategic IT initiatives that support the business and respond more quickly to changing user requirements. At the same time risks of database downtime, which previously resulted from manual configuration errors, are significantly reduced.

As an example, management operations requiring rolling restarts of a MySQL Cluster database that previously demanded 46 manual commands[2] and which consumed 2.5 hours of DBA time[3] can now be performed with a single command, and are fully automated, serving to reduce:

- Cluster management complexity and overhead.
- Custom scripting of management commands.
- The risk of downtime through the automation of configuration and change management processes.

You can learn more about MySQL Cluster Manager by browsing to www.mysql.com/cluster/mcm

MySQL Support, Consulting and Training

MySQL Support

Oracle offers 24x7, global support for MySQL. The MySQL Support team is composed of seasoned MySQL developers, who are database experts and understand the issues and challenges you face. With Oracle Premier Support, you can lower the total cost and risk of owning your MySQL databases, improve the return from your IT investment, and optimize the business value of your IT solutions. MySQL support is included in the subscription for end users, and available separately from commercial licenses for ISVs and OEMs. Oracle Premier Support for MySQL include the following features:
• 24 x 7 production support
• Unlimited support incidents
• Knowledge Base
• Maintenance releases, bug fixes, patches and updates
• MySQL consultative support

You can learn more at http://www.mysql.com/support/

MySQL Consulting

Oracle offers a full range of MySQL consulting services. From Architecture and Design for new projects, through Performance Tuning to optimize existing applications, or migrating from an alternative database to MySQL, we have an affordable solution for you.

You can learn more at http://www.mysql.com/consulting/

MySQL Training

Oracle offers a comprehensive set of MySQL training courses that give you a competitive edge in building world-class database solutions. Designed for DBAs and Developers of all levels, covering both database and application design, including courses dedicated to the development of dynamic

[2] Based on a MySQL Cluster configuration comprised of 4 x MySQL Cluster Data Nodes, 2 x MySQL Server SQL Nodes and 2 x MySQL Cluster Management Nodes implemented across individual servers (8 x total). Total operation comprised the following commands: 1 x preliminary check of cluster state; 8 x ssh commands per server; 8 x per-process stop commands; 4 x scp of configuration files (2 x mgmd & 2 x mysqld); 8 x per-process start commands; 8 x checks for started and re-joined processes; 8 x process completion verifications; 1 x verify completion of the whole cluster. Total command count does not include manual editing of each configuration file.

[3] Based on a DBA restarting 4 x MySQL Cluster Data Nodes, each with 6GB of data, and performing 10,000 operations per second http://www.clusterdb.com/mysql-cluster/mysql-cluster-data-node-restart-times/

web applications.
http://www.mysql.com/training/

Conclusion

Designing, managing and scaling web-based infrastructure is hard. Technology innovations are happening at a much greater pace than ever before, and user requirements change even more rapidly.

The demand for new services, coupled with the explosion in data that needs to be managed creates significant obstacles for any organization looking to use the web to drive their business.

By leveraging the best practices identified in this whitepaper, we hope to have at least provided a starting point to enable you to build the next web phenomenon.

Additional Resources

MySQL Enterprise: Database. Monitoring. Support.
http://www.mysql.com/why-mysql/white-papers/mysql_wp_enterprise_ready.php

MySQL Query Analyzer: Overview
http://www.mysql.com/why-mysql/white-papers/mysql_wp_queryanalyzer.php

Whitepaper - MySQL Cluster for Web & ecommerce Applications:
http://www.mysql.com/why-mysql/white-papers/mysql_wp_Cluster_For_OnlineApps.php

Designing and Implementing Scalable Applications with Memcached and MySQL
http://www.mysql.com/why-mysql/white-papers/mysql_wp_memcached.php

Scaling Web Services with MySQL Cluster: An Alternative Approach to MySQL & memcached
http://www.mysql.com/why-mysql/white-papers/mysql_wp_cluster_ScalingWebServices.php

On-Demand Webinar – 5 Easy Steps to Getting Started with MySQL Cluster:
http://www.mysql.com/news-and-events/on-demand-webinars/display-od-566.html